# Lab 4: Bytecode executor

February 28, 2023

The goal of this lab is to develop a bytecode executor for a stack based virtual machine for the *kl* interpreter.

## 1 Get the code

The base code used in the labs is in the directory */courses/TDDE55/Labs/kl*.

To get the code and start working on it, in your repository:

```
cp -r /courses/TDDE55/Labs/kl .
```

This will copy the skeleton for the *kl* interpreter, you can now find it in the directory *kl*. In the rest of the document, we will refer to this directory as *dir_to_kl*.

## 2 Instruction set

The instruction set for the bytecode executor is defined in the file *klib/bytecode/OpCodes.py*. It is composed of 25 instructions, divided in 6 groups:

- *Module*
- *Stack Manipulation*
- *Environment and Objects Manipulation*
- *Control*
- *Exceptions*
- *Function Creation*

Carefully study the file side-by-side with the tests (in `/courses/TDDE55/Labs/Lab4/Tests/test_executor.py`).

## 3 Bytecode Executor

In this lab you will be developing the bytecode executor. Before starting, you should have a look at some of the existing class in the klib directory.

- `klib.bytecode.Program` it contains a list of instruction defining a code that can be executed, be it a program or a function
- `klib.bytecode.Instruction` it defines a single instruction for a `Code` object, it contains the instruction number and possible arguments

- `klib.bytecode.OpCodes` as mentioned in the previous section, it defines the instruction set
- `klib.environment.*` it defines the classes used for an Environment, it is similar to the Environment class that you have used in the previous labs.
- `klib.interpreter.*` it defines some classes used to implement the interperter, such as support for modules
- `klib.native.*` it defines native modules, or some modules that can be used to call python function from kl programs
- `klib.vm.Stack` it defines the stack that is use by the virtual machine.
- `klib.parser` the parser for the kl language, it is similar, but with extension, to the parser you implemented in the exercises.
- `klib.io` Input/Output library

The class *klib.vm.Executor* (in *klib/vm/executor.py*) is the class that you should implement in this lab.

# 4 Run the executor unit tests

You can run the test suite for the executor with the following command:

```
1 tdde55_lab4_tests dir_to_kl executor
```

# 5 KL tests suite

## 5.1 Test suite

You can find the test suite in */courses/TDDE55/Labs/Lab4/Tests/lang*. Each test case is made out of a file that needs to be interpreted by your code.

There are currently eight categories of tests:

1. *001_function_call.kl*: test for the assert function
2. *002_function_call_fail.kl*: negative test for the assert function (if you run this file separately, you should get an exception!)
3. *003_function_no_return.kl*: test for calling a function with no return
4. *010_basic_expressions.kl*: tests for basic expression (addition, multiplication, logical...)
5. *011_basic_expressions_function.kl*: tests for basic expression (addition, multiplication, logical...) in a function body
6. *012_define.kl*: test for definition of values
7. *013_define_fail.kl*: test failure of assignment of a value
8. *014_define_function.kl*: test for definition of values in a function body
9. *015_define_function_fail.kl*: test failure of assignment of a value in a function body

10. *020_cell.kl*: test for the creation and use of cells
11. *021_cell_function.kl*: test for the creation and use of cells in a function body
12. *022_sub_environment.kl*: test for sub environment
13. *023_sub_environment_function.kl*: test for sub environment in a function body
14. *024_cell_expressions.kl*: test for the use of cells in expressions
15. *025_group.kl*: test for grouping of expressions
16. *026_group_function.kl*: test for grouping of expressions in a function body
17. *027_reference.kl*: test for the use of reference
18. *030_cond_expressions.kl*: the for conditional expression
19. *031_cond_expressions_function.kl*: the for conditional expression in a function body
20. *060_exceptions.kl*: test for exceptions
21. *061_exceptions_function.kl*: test for exceptions in a function body
22. *900_factorial.kl*: test for factorial

## 5.2 Testing your interpreter

You can run the test suite for the interperter with the following command:

```
tdde55_lab4_tests dir_to_kl test_suite
```

Since running the full test suite can take a bit of time, you can run a individual kl script with:

```
dir_to_kl/kl [scriptname]
```

The complete test suite is in `/courses/TDDE55/Labs/Lab4/Tests/`.
Finally before submitting your result, you can check all tests with:

```
tdde55_lab4_tests dir_to_kl
```